第2章 FPGAにマイクロプロセッサを接続して制御能力を高める

ADuC7026インターフェース回路の設計

江崎雅康

画像ベースボードの狙いは，各種画像フィルタや高速テンプレート・マッチング，モーション検出など高度な画像処理アルゴリズムを検証するためのハードウェア・プラットホームを準備することである．そこで，制御用にマイクロプロセッサを用意した．今回は，A-D/D-Aコンバータを内蔵したADuC7026を採用した．（編集部）

1 マイクロプロセッサADuC7026を選んだ理由 …アドレス/データ時分割多重化バスで信号線が少ない

本誌2007年8月号で紹介した画像ベースボードCQ-SP3EDWの設計に当たって，いちばん苦慮した点は付属基板の信号ピンのやりくりです．

付属基板に搭載されたXC3S250E-VQ100は100ピン・パッケージです．電源を除くとコネクタ経由で使える信号線の数は，次のように限られています．

- 入出力ピン　53本
- 入力専用ピン　4本
- クロック　1本

16ビット・バス幅のVGA画像フレーム・メモリを接続するためには最低，

- データ線　16本
- アドレス線　18本
- 制御線　3本（/CS，/RD，/WR）

の計37本を必要とします．しかし，その余裕はないので，16ビット・バス幅のVGA画像フレーム・メモリの搭載は断念しました．

制御用マイクロプロセッサは熟慮の末，ADuC7026（米国Analog Devices社）を採用しました．このLSIは，本誌2006年3月号の付属企画に採用されたARMコア内蔵のマイクロプロセッサです．

ADuC7026はアドレス線とデータ線が多重化されたマルチプレクス・バスになっています．図1に示すように，

- アドレス/データ信号線　16本（AD0～AD15）
- 制御信号線　4本（nMS0，nRD，nWR，AE）

の計20本で，FPGA内のレジスタやメモリ・ブロックをアクセスできます．

ADuC7026は水晶発振子（32.768kHz）のクロックを，内蔵しているPLL（Phase-locked Loop）回路で逓(てい)倍して得られるクロック（最高41.78MHz）で動作します．

図2の機能ブロック図に示すように，

- フラッシュ・メモリ　62Kバイト
- SRAM　8Kバイト
- 12ビットA-Dコンバータ　12＋4チャネル
- 12ビットD-Aコンバータ　4チャネル
- GPIO（General Purpose Input Output）
- 外部メモリ・インターフェース

などを備えたマイクロプロセッサです．

画像ベースボードは図1に示すように，USB-シリアル変換ICのCP2102（米国Silicon Laboratories社）を搭載しています．ADuC7026のシリアル・ポートをUSBに変換し，仮想COMポートとしてプログラムのダウンロードおよび

KeyWord　ADuC7026，VGA画像フレーム・メモリ，ARMコア，マルチプレクス・バス，リード・タイミング，ライト・タイミング

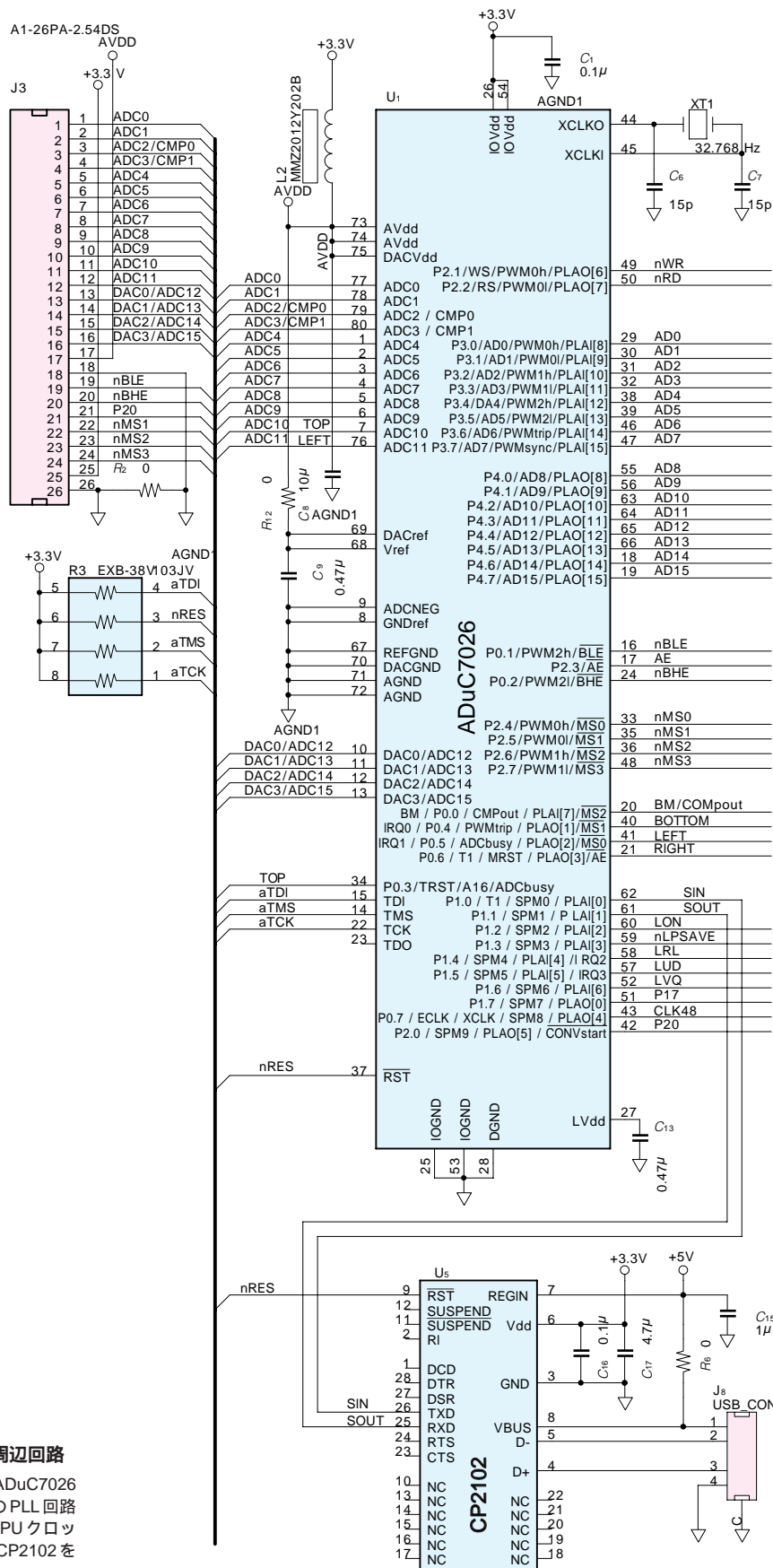



図1
ADuC7026マイクロプロセッサ周辺回路

ARMコア内蔵マイクロプロセッサADuC7026は32.768kHzの発振周波数を内蔵のPLL回路により最高41.78MHzに逓倍してCPUクロックとする．シリアル-USB変換ICのCP2102を画像ベースボード上に搭載している．

汎用シリアル通信ポートとして使うことができます．

電源は基板上に降圧レギュレータを搭載しました．図3に示す回路でDCジャックに供給される5V電源を3.3Vに変換します．同期整流方式の高効率回路であり，3A程度の負荷電流を供給できます．

付属基板上には可変出力型3端子レギュレータ「LM317」を取り付ける配線パターンが用意されています．しかし，液晶モジュール，CMOSカメラ・モジュール，そして将来的には画像フレーム・メモリを追加することを考えると，電流容量が不足します．

そこで画像ベースボード上にこの3A程度の負荷電流に耐える電源を備えました．5VのDCアダプタをDCジャックJ_{11}に接続することにより，電源を供給します．

2 マイクロプロセッサ・インターフェース回路の全体構成

ADuC7026のプログラムで，付属FPGA基板上のLEDを点滅させてみましょう．図4は回路のブロック図です．

XC3S250E内部にLED制御レジスタを設けます．ADuC7026はこのレジスタの内容を書き換えることにより，XC3S250Eの98ピンに接続されたLEDを点滅させます．

アドレス/データ・バスの幅は16ビットで，アドレスとデータは時分割により送受信されます．アドレス・ラッチ・イネーブル（AE）は多重バスからアドレス信号を切り出すためのラッチ信号です．

ADuC7026は次に示す制御信号である，

- /WR　書き込みストローブ

図2 マイクロプロセッサADuC7026の機能ブロック図
12ビット16チャネルのアナログ入力，12ビット4チャネルのD-Aコンバータ出力端子を備える．最高512Kバイトの外部メモリを接続可能．

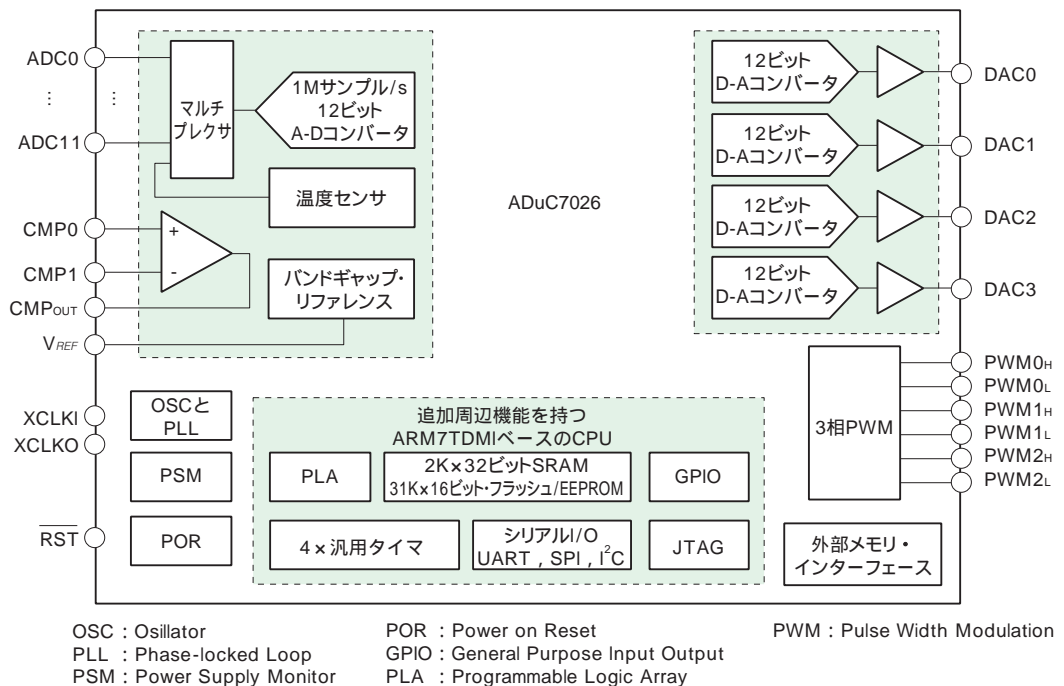
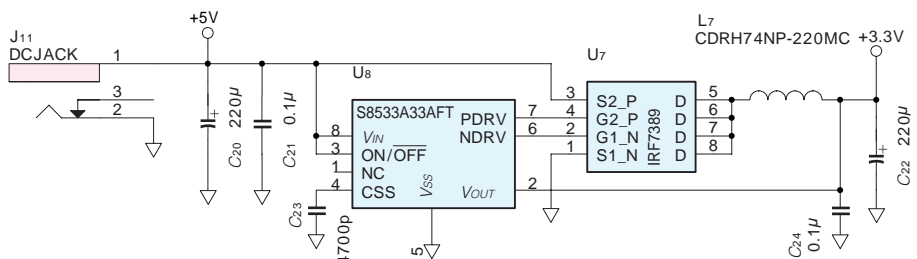
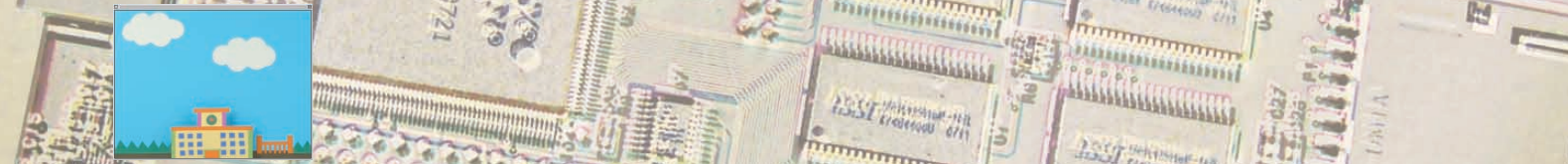



図3 同期整流回路を使った降圧レギュレータ回路
同期整流方式による高効率降圧レギュレータ．最大3Aの3.3V出力が可能．

●/RD　読み出しストローブ
●/MS0　メモリ・セレクト信号

により，メモリ空間に配置されたXC3S250E内部のレジスタにアクセスします．

3 マイクロプロセッサ・インターフェース回路

XC3S250E内部のマイクロプロセッサ・インターフェース回路は，ADuC7026の外部メモリ空間0のアクセス信号により，指定レジスタのデータをリード/ライトします．

バイト・イネーブル信号BEN0，BEN1はFPGAに接続されていないため使用できません．従って，データのリード・ライトは必ずワード（16ビット）単位となります．

FPGA内部にレジスタを4ワード（8バイト）用意し，ADuC7026の外部メモリ空間アドレス0x10000000から4ワードに割り付けます．レジスタ・アクセスはADuC7026の内部クロック41.78MHzに同期して行われます．

ADuC7026のリード・サイクルのタイミング・チャートを図5に，ライト・サイクルのタイミング・チャートを図6に示します．メモリ・セレクト（/MS0）は，リード/ライト・サイクルの開始から終了までアサートされます．外部メモリを使用する場合は信号/MS0をメモリのチップ・セレクト（/CS）へ接続します．

アドレス・ラッチ・イネーブル（AE）はバスへアドレス

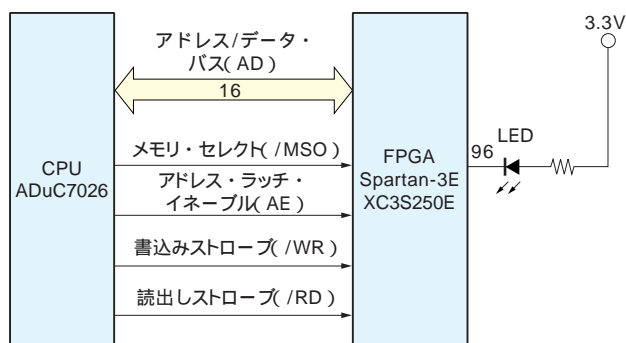



図4　マイクロプロセッサ・インターフェース回路のブロック図
ADuC7026はアドレス・バスとデータ・バスが多重化されているので，20本の信号線で外部メモリ空間のアクセスができる．FPGAの内部レジスタのアクセスをLEDの点滅で確認する．

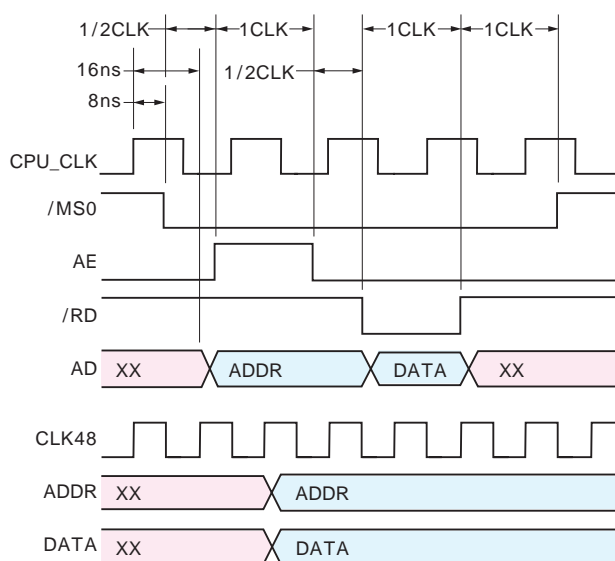
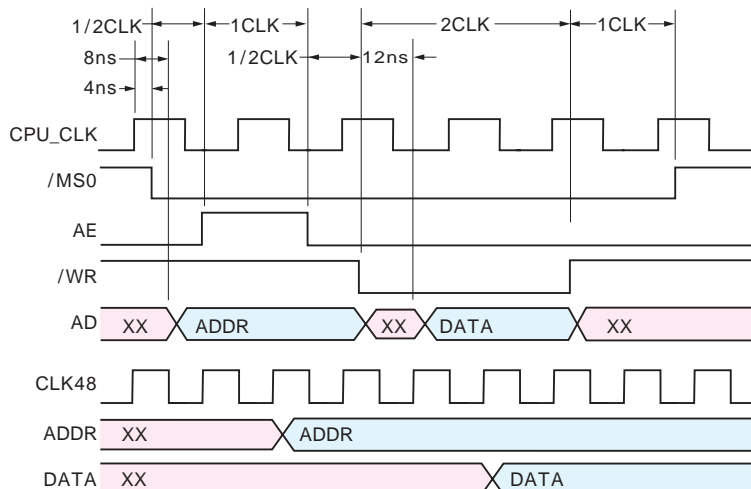



図5　リード・サイクルのタイミング・チャート
マイクロプロセッサ（ADuC7026）は41.78MHz，FPGAは48MHzと異なるクロックを使っているので，タイミング設計には注意を要する．アドレスとデータは時分割多重化されてADバスに乗る．

図6
ライト・サイクルのタイミング・チャート
ADuC7026の初期設定では，タイミング上データを正しく書き込めない恐れがある．XMOPARレジスタの設定変更により，WRがアサートされる期間を1クロック分増やした．

を出力している間アサートされます．

書き込みストローブ(/WR)はライト・データを出力している間アサートされます．読み出しストローブ(/RD)はリード・データの入力を待っている間アサートされます．

アドレス/データ・バス(AD)は，アドレスおよびデータが流れる双方向のバスです．**リスト1**に本回路のVHDLソースを示します．

4 リード・サイクルのタイミング設計

リード・サイクルのタイミング・チャート(**図5**)で，CPU_CLKはADuC7026の内部クロックであり，外部には出力されません．

CLK48は，XC3S250Eが内部で使用している48MHzのクロックです．使用しているクロックが異なるため，アド

リスト1　マイクロプロセッサ・インターフェース回路のVHDLソース

```
--------------------------------------------------------
-- Design Name:     reg_rw_led
-- Module Name:     reg_rw_led - Behavioral
--------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity reg_rw_led is
port(
      clk48_in            : in     std_logic;
      nMS                 : in     std_logic;
      AE                  : in     std_logic;
      nWR                 : in     std_logic;
      nRD                 : in     std_logic;
      AD                  : inout  std_logic_vector(15
                            downto 0);
      nLEDO               : out    std_logic;

      dummy               : in     std_logic
);
end reg_rw_led;

architecture Behavioral of reg_rw_led is

      signal clk48        : std_logic;
      signal s_reset      : std_logic;
      signal s_locked48   : std_logic;
      signal reg0         : std_logic_vector(15 downto 0);
      signal reg1         : std_logic_vector(15 downto 0);
      signal reg2         : std_logic_vector(15 downto 0);
      signal reg3         : std_logic_vector(15 downto 0);
      signal s_addr       : std_logic_vector(AD'range);
      signal s_sel        : std_logic_vector(1 downto 0);
      signal s_data       : std_logic_vector(AD'range);
      signal s_nWR        : std_logic;

      COMPONENT dcm_clk48
      PORT(
               CLKIN_IN : IN std_logic;
               RST_IN : IN std_logic;
               CLKIN_IBUFG_OUT : OUT std_logic;
               CLK0_OUT : OUT std_logic;
               LOCKED_OUT : OUT std_logic
      );
      END COMPONENT;

begin

      -- Generate clock
      U_DCM_CLK48 : dcm_clk48
      PORT MAP(
               CLKIN_IN  => clk48_in,
               RST_IN             => '0',
               CLKIN_IBUFG_OUT => open,
               CLK0_OUT  => clk48,
               LOCKED_OUT         => s_locked48
      );
      s_reset   <= not s_locked48;

      s_sel     <= s_addr(1 downto 0);
      s_data    <=          reg0 when s_sel = "00" else
                            reg1 when s_sel = "01" else
                            reg2 when s_sel = "10" else
                            reg3;

      -- Register Read
      AD        <= s_data when (nMS = '0' and nRD = '0')
                            else (others => 'Z');

      -- Address Latch
      process (clk48, s_reset)
      begin
               if (s_reset = '1') then
                        s_addr   <= (others => '0');

               elsif (rising_edge(clk48)) then
                        if (nMS = '0' and AE = '1') then
                                 s_addr   <= AD;
                        end if;

               end if;
      end process;

      -- Register Write
      process (clk48, s_reset)
      begin
               if (s_reset = '1') then
                        reg0     <= (others => '0');
                        reg1     <= (others => '0');
                        reg2     <= (others => '0');
                        reg3     <= (others => '0');
                        s_nWR    <= '1';

               elsif (rising_edge(clk48)) then
                        s_nWR    <= nWR;
                   if (nMS = '0' and s_nWR = '0' and nWR
                                                = '0') then
                        case s_sel is
                        when "00" =>
                                      reg0    <= AD;
                        when "01" =>
                                      reg1    <= AD;
                        when "10" =>
                                      reg2    <= AD;
                        when "11" =>
                                      reg3    <= AD;
                        when others =>
                                      null;
                        end case;
                   end if;

               end if;
      end process;

      -- LED Output
      nLEDO     <= reg0(0);


end Behavioral;
```

リスト2　LEDの点滅のプログラム

```
****************************************************
      LED Blink for CQ-SP3EDW
      Build By KEIL uVISION3
****************************************************

#include <ADuC7026.H>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>

#define FPGA_REG        0x10000000
#define CR                        0x0D

#define true                      1
#define false                     0
typedef int bool;

typedef   signed char int8_t;
typedef   signed short int16_t;
typedef   signed long int32_t;
typedef unsigned char uint8_t;
typedef unsigned short uint16_t;
typedef unsigned long uint32_t;

static bool timer_interrupted = false;

static void uart_putchar(int8_t ch)
{
      while (0x20 != (COMSTA0 & 0x20)) {}

      if (ch == '\n')  {
              COMTX = CR;        // output CR
      } else {
              COMTX = ch;
      }
}

static int8_t uart_getchar(void)
{
      while (!(0x01 == (COMSTA0 & 0x01))) {}
      return (int8_t)COMRX;
}

static void uart_putstring(const int8_t *s)
{
      while (*s != 0) {
              uart_putchar(*s++);
      }
}

static void uart_printf(const int8_t *fmt, ...)
{
      static int8_t uart_buf[128];
      va_list args;

      va_start(args, fmt);
      vsprintf(uart_buf, fmt, args);
      uart_putstring(uart_buf);
      va_end(args);
}

static void waitfor_msec(int msec)
{
      int i;

      for (i = 0; i < msec / 10; i++) {
              while (!timer_interrupted) {}
              timer_interrupted = false;
      }
}

void IRQ_Handler(void) __irq
{
      timer_interrupted = true;
      T1CLRI = 1;
}

static void write_mem(volatile unsigned short *addr, unsigned
short data)
{
      uart_printf("write addr:%08X data:%04X\n", addr, data);
      *addr = data;
}

static uint16_t read_mem(volatile unsigned short *addr)
{
      unsigned short data;

      data = *addr;
      uart_printf("read addr:%08X data:%04X\n", addr, data);
      return data;
}

static void init_hardware(void) ◀──────── init_hardware関数
{
      // PLL
      PLLKEY1   = 0x000000aa;      // Release PLLKEY
      PLLCON    = 0x00000021;      // Uset external 32kHz
      PLLKEY2   = 0x00000055;      // Set PLLKEY
      // Clock select
      POWKEY1   = 0x00000001;      // Release POWKEY
      POWCON    = 0x00000000;      // Active mode,41.78MHz
      POWKEY2   = 0x000000f4;      // Set POWKEY
      // LED
      GP0CON    = 0x00000000;
      GP0PAR    = 0x20000000;
      GP1DAT    = 0xFF000000;      // P1.7 Output '0'
      GP1DAT    = 0xFFE40000;
      // Serial
      GP1CON    = 0x00000011;      // SOUT,SIN
      COMCON0   = 0x00000080;      // COMDIV0/1 enable
      COMDIV0   = 0x00000088;      // 9600bps
      COMDIV1   = 0x00000000;      // 9600bps
      COMCON0   = 0x00000007;      // TX/RX enable
      // External memory
      GP2CON    = 0x00022220;      // nMS0,AE,nRD,nWR
      GP3CON    = 0x22222222;      // AD7-AD0
      GP4CON    = 0x22222222;      // AD15-AD8
      XMCFG     = 0x00000001;      // External memory enable
      XM0CON    = 0x00000003;      // 16bit bus, XM0 enable
      XM0PAR    = 0x00008710;      // WR enable, nWR strobe 2CLK
      // Timer
      T1LD = 13;                // 10msec, 41780000/32768/T1_Freq
      T1CON = 0xCF;             // start Timer1
      IRQEN = 0x00000008;       // Timer1 IRQ enable
}

int main (void)
{
      uint16_t data;

      init_hardware();
      waitfor_msec(5000);

      uart_printf("program start!!\n");

      for (;;) {
              GP1DAT ^= 0x00800000;      // Reverse P1.7
              data = read_mem(FPGA_REG + 0);
                                         // Read FPGA register
              data = ~data;              // Reverse register
              write_mem(FPGA_REG + 0, data);
                                         // Write FPGA register
              waitfor_msec(500);
      }

      return 0;
}
```

表1
ADuC7026レジスタ設定値
ADuC7026はメモリ・マップド・レジスタ(MMR)により，ピン機能の設定，外部バス幅，アクセス・タイミングなどの設定を行う．

レジスタ	アドレス	初期値	設定値	意　味
GP2CON	0xFFFFF408	0x00000000	0x00022220	/MS0，AE，/RD，/WR有効
GP3CON	0xFFFFF40C	0x00000000	0x22222222	AD7-AD0有効
GP4CON	0xFFFFF410	0x00000000	0x22222222	AD15-AD8有効
XMCFG	0xFFFFF000	0x00000000	0x00000001	外部メモリ空間有効
XM0CON	0xFFFFF010	0x00000000	0x00000003	外部メモリ空間0有効，バス幅16ビット
XM0PAR	0xFFFFF020	0x000070FF	0x00008710	/WRを1クロック延長

レスおよびデータの取りこぼしに注意する必要があります．

今回の回路ではCLK48のほうがCPU_CLKよりも周波数が高いため，AEおよびストローブが1クロック間だけアサートされていても取りこぼすことはありません．

本当はCPU_CLKの周期(約24ns)とCLK48の周期(約21ns)の違いはわずかですが，**図5**では分かりやすくするために誇張して表現しています．

リード・サイクルが開始されると/MS0がアサートされます．その1/2クロック後にAEがアサートされます．AEがアサートされている間はアドレスがバスに出力されているので，CLK48の立ち上がりでADを取り込みます．

アドレスが決定した時点で，組み合わせ回路によりレジスタを選択します．選択されたレジスタのデータは，/RDがアサートされている間，ADへ出力します．

5 ライト・サイクルのタイミング設計

ライト・サイクルのタイミング・チャートは**図6**です．

ライト・サイクルが開始されると，/MS0がアサートされます．その1/2クロック後に，AEがアサートされます．

リード・サイクルと同様に，CLK48の立ち上がりでADを取り込みます．次にライト・データを取り込む必要がありますが，ここで一つ問題があります．

データ・シートによると，/WRがアサートされてから最大12ns後にライト・データが確定するとあります．初期設定では/WRがアサートされる期間は1クロック間，すなわち約24nsです．

つまり，/WRがアサートされている期間のうち，ライト・データが確定している期間は最小で24－12＝12nsとなります．これはCLK48の1周期の約21nsを下回るため，正しいデータを取り込めない可能性があります．

このためADuC7026のXM0PARレジスタの設定により，/WRをアサートする期間を1クロック分だけ増やしました．CLK48にて/WRをラッチし，/WRが2クロックの間“0”であった場合，その時点のADをライト・データとして取り込み，選択されたアドレスのレジスタに書き込みます．

6 ADuC7026の設定

リスト2は付属基板上のLEDを点滅させるプログラムです．開発はKeil社(現在はARM社)の統合開発環境μvision3(本誌2006年3月号付属CD-ROMに収録)を使いました．

リスト2の関数，`init_hardware`はADuC7026の初期化を行います．ADuC7026は外部メモリ空間を有効にするために内部レジスタを設定する必要があります．各レジスタの設定値を**表1**に示します．

まず外部メモリ空間0を有効にします．外部メモリ空間0は，アドレス0x10000000から0x1000FFFFの64Kバイトの空間と定められています．

FPGA内部のレジスタに書き込むための関数として，

- `write_mem`　FPGA内部レジスタに書き込む
- `read_mem`　FPGA内部レジスタを読み出す

を用意しました．デバッグを容易にするためにアクセス・アドレスとデータをUARTに出力します．このメッセージは，画像ベースボードとUSB接続したパソコンのハイパータームに表示できます．

アドレス0x10000000のビット0に，1を書き込むとLEDが消灯し，0を書き込むとLEDが点灯します．プログラムは10ms間隔のタイマ割り込みを使用して500msごとに点灯と消灯を繰り返します．

参考・引用*文献

(1)*アナログ・デバイセズ；高精度アナログ・マイクロコントローラADuC7019/20/21/22/24/25/26/27データシート，2006年.

えさき・まさやす
(株)イーエスピー企画
代表取締役